Highly Parallel Hardware-oriented Algorithm for Jacobi SVD of Hermitian Quaternion Valued Matrix

Evgueni Doukhnitch*1, Vadim Podbelskiy2

1Department of Computer Engineering, Istanbul Aydin University/Istanbul, Turkey
2Department of Sofware Engineering, National Research University Higher School of Economics/Moscow, Russia
*1evgenydukhnich@aydin.edu.tr; 2vpodbelskiy@hse.ru

Abstract

In this study, new highly parallel algorithm of two-sided Jacobi 8-D transformation is suggested. It is oriented on VLSI-implementation of special processor array. This array is built using 8-D CORDIC algorithm for quaternion valued matrix singular value decomposition. Accuracy analysis and simulation results are added. Such array can be utilized to speed up the Jacobi method realization to compute the SVD of a quaternion matrix in signal and image processing.

Keywords

Quaternion Matrix; Jacobi Method; SVD; Processor Arrays

Introduction

Singular value decompositions (SVD) and eigenvalue decomposition (EVD) are important problems in linear algebra and digital signal processing. The most widely employed method of solving SVD problem is zeroing the entries in a matrix by a sequence of rotations or reflections (see Rader [1996]). In many such computations, it is necessary to vanish some elements selectively with matrix similarity transformation. High practical importance of these problems leads to the need of designing hardware-oriented algorithms for their fast VLSI implementation. The Jacobi method is the frequently used tool and very high throughputs are required from the special processors which will be used as array's elements for the parallel computation of these transformations.

The coordinate rotation computer (CORDIC) is an effective processing element for such systolic arrays which executes these transformations without multiplications (see Meher [2009]). The original CORDIC algorithm processes two real numbers at a time. It is well below the needed throughput for DSP. To speed up the matrix computations, they use higher dimensional rotations. To generalize the original CORDIC algorithms, Hsiao and Delosme offered the Quaternion or Pseudo-quaternion CORDIC algorithms for 4-D rotations(see Hsiao [1996]). These algorithms are used for parallel decomposition of complex matrices and demonstrate significant speed up in contrast to the original 2-D CORDIC. Following this direction, Octonion CORDIC algorithm (OCA) for 8-D rotations was suggested in Doukhnitch [2002] and generalized in Doukhnitch [2011] to implement Givens rotation applied to quaternion valued vectors.

The study of quaternion matrices has gained interest in many practical areas in recent years (see Bihan, Miron, Zhang). It is motivated with smaller complexity in terms of storage and computations needed for a direct quaternion algorithm (seeBihan [2007]). Besides that, use of quaternion operands for computations leads to less computational errors. The application of quaternions to represent color images has been introduced by Sangwine [1996]. A color image is represented as a pure quaternion image:

$$S(x,y) = r(x,y)\boldsymbol{i} + g(x,y)\boldsymbol{j} + b(x,y)\boldsymbol{k}, \quad (1)$$

where*r(x,y)*, *g(x,y)*, *b(x,y)* are respectively the red, green and blue components of a pixel at position *(x,y)* in the image *S(x,y)*. This representation has allowed the definition of powerful tools for color image processing such as Fourier transforms, compression, correlation, recognition or edge detection including using SVD (see Bihan[2003], Pei [2003]). In the last few years, quaternions were used in statistical signal processing for multiple sources characterization and discrimination.

In accordance with Jacobi method, they divide the rows of a given matrix into pairs and for each pair they calculate an angle for two-sided rotation to null off-diagonal element. In 1983, the Brent-Luk-Van Loan (BLV) systolic array was proposed for fast SVD parallel computation, see Brent [1983]. In Cavallaro [1988], there was suggested to use CORDIC processors

for such array. B.Yang and J.F. Bohme proposed the Two Plane Rotation (TPR) method to perform two-sided 2×2 SVD as two parallel rotations, see Yang [1991]. Hsiao and Delosme proposed to use multidimensional CORDIC algorithms in systolic array for both real and complex matrices (see Hsiao [1996]). There are many modifications of Jacobi method realization in publications. For example, the approximate CORDIC-based Jacobi algorithm was considered in Gotze [1993]. In Strumpen [2003], a stream SVD algorithm was suggested. In Snopce [2010], a preliminary complex-to-real transformation was proposed.

H. C. Lee and then F. Zhang [1997] suggested calculating SVD of quaternion matrices by means of the SVD of its equivalent complex matrix with twice size (see also Bihan [2003], Pei [2003]). The two-sided Jacobi algorithm has also been used to solve SVD of a quaternion matrix directly without such decomposition (see Bihan [2007]).

In this paper, we suggest to use Octonion CORDIC algorithm for hardware realization of Jacobi transformation with BVL processor array. The only difference is in new processor elements (PEs). They can speed up the process because of parallel calculation of elements of 8-D vectors (pairs of quaternions) and alignmentat runtime of angle calculation and rotations implementation. But it needssignificant modification of diagonal PEs. It is due to the fact that OCA does not have an evident representation of angle for 8-D rotation and TPR method cannot be realized directly.

In Section 1, a short description of Givens transformation with OCA for quaternion operands is given. In Section 2 OCA modification for Jacobi and TPR methods is described. The algorithm of SVD implementation with processor array is suggested in Section 3. In Section 4, the error analysis together with simulation results are shown. Finally, in the last section, the conclusions are given.

Quaternion Vector Transformation with OCA

The typical matrix operation for a matrix decomposition is an one-sided linear transformation of rotation:

$$\boldsymbol{Y}=\boldsymbol{P}*\boldsymbol{X} \tag{2}$$

If we have a quaternion valued vector $\boldsymbol{X}=(\boldsymbol{q}_1, \boldsymbol{q}_2)^T$ in (2) we can describe an 8-D rotation of equivalent 8-D real vector $\boldsymbol{X}=(q_{11}, q_{12}, q_{13}, q_{14}, q_{21}, q_{22}, q_{23}, q_{24})^T$, where quaternion l has components $\boldsymbol{q}_l=(q_{l1}, q_{l2}, q_{l3}, q_{l4})$ (l=1,2).

The Cayley numbers(see Ward [1997])or *octonions* (8-D objects)can be used to represent 8x8 matrix P as a product of octonions of elementary rotations:

$$\boldsymbol{P}=\prod\nolimits_{\boldsymbol{i}=\boldsymbol{0}}^{\boldsymbol{n}}\boldsymbol{M}_{\boldsymbol{i}} \tag{3}$$

These octonions have a unit norm (N_M=1), and can be represented in polar form:

$$\boldsymbol{M}_i=\cos\varphi_i+\sin\varphi_i\,(\alpha\boldsymbol{i}+\beta\boldsymbol{j}+\gamma\boldsymbol{k}+\delta\boldsymbol{l}+\lambda\boldsymbol{q}+\mu\boldsymbol{r}+\rho\boldsymbol{s}) \tag{4}$$

where $\varphi_i=atan2^{-i}$. Following Doukhnitch [2002], the 8-D rotation matrix can be represented as $\boldsymbol{R}_{8,i}=(1/\cos\phi_i)\boldsymbol{M}_i$or

$$R_{8,i}=\begin{bmatrix} 1 & \alpha_i t_i & \beta_i t_i & \gamma_i t_i & \delta_i t_i & \lambda_i t_i & \mu_i t_i & \rho_i t_i \\ -\alpha_i t_i & 1 & -\delta_i t_i & -\rho_i t_i & \beta_i t_i & -\mu_i t_i & \lambda_i t_i & \gamma_i t_i \\ -\beta_i t_i & \delta_i t_i & 1 & -\lambda_i t_i & -\alpha_i t_i & \gamma_i t_i & -\rho_i t_i & \mu_i t_i \\ -\gamma_i t_i & \rho_i t_i & \lambda_i t_i & 1 & -\mu_i t_i & -\beta_i t_i & \delta_i t_i & -\alpha_i t_i \\ -\delta_i t_i & -\beta_i t_i & \alpha_i t_i & \mu_i t_i & 1 & -\rho_i t_i & -\gamma_i t_i & \lambda_i t_i \\ -\lambda_i t_i & \mu_i t_i & -\gamma_i t_i & \beta_i t_i & \rho_i t_i & 1 & -\alpha_i t_i & -\delta_i t_i \\ -\mu_i t_i & -\lambda_i t_i & \rho_i t_i & -\delta_i t_i & \gamma_i t_i & \alpha_i t_i & 1 & -\beta_i t_i \\ -\rho_i t_i & -\gamma_i t_i & -\mu_i t_i & \alpha_i t_i & -\lambda_i t_i & \delta_i t_i & \beta_i t_i & 1 \end{bmatrix} \tag{5}$$

The scaling factor is $k_i=1/\cos\phi_i=\sqrt{1+7t_i^2}$. Many strategies were proposed to force $k=\prod_i k_i$ to be a simple signed bits representation for efficient scaling correction, e.g. Yang [1991] or Cavallaro [1988]. All of them use additional sequence of shift-add operations with average total number$n/4$.

The rotation parameters t_i are equal to $2^{-f(i)}$ where $\{f(i)\}$ is a non-decreasing positive integer sequence:

$$f(i)=\{0,1^*,2^*,3^*,4^*,5,6,7^*,8,\ldots,13^*,14,\ldots,n\} \tag{6}$$

where the sign * denotes a repetition. Thus, for n-bits accuracy (n>13), the total number of iterations is $(n+6)$ with guaranteed convergence (see Doukhnitch [2011]).

The control signs $\alpha_i...\rho_i$ are either 1 or-1. Without performing the scaling by k_i^{-1}, the implementation of one elementary rotation consists of eight concurrent shift-and-add operations. The Octonion CORDIC algorithm:

$$\boldsymbol{Y}_{i+1}=\boldsymbol{R}_{8,i}\,\boldsymbol{Y}_i,\ (\overline{i=0,n}\,;\ \boldsymbol{Y}_0=\boldsymbol{X}), \tag{7}$$

as well as original CORDIC algorithm can be used in two modes–rotation (*application* of transformation) and vectoring (*evaluation* of parameters of transformation). The usual task of vectoring in such algorithms is the annihilation of required components in a given vector, as after Volder's plane rotation, one of the two components becomes zero, while in 8-D space, seven components may be zero. A sequence of elementary rotations with matrices (5) is applied to an 8-D vector $\boldsymbol{X}=[x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8]^T$ to bring it along the first canonical axis. To achieve this, the control signs are

selected for (5) according to the following expressions:

$$\begin{aligned} \alpha_i &= f_i \cdot \mathrm{sign}(z_{2,i}); \ \beta_i = f_i \cdot \mathrm{sign}(z_{3,i}); \\ \gamma_i &= f_i \cdot \mathrm{sign}(z_{4,i}); \ \delta_i = f_i \cdot \mathrm{sign}(z_{5,i}); \\ \lambda_i &= f_i \cdot \mathrm{sign}(z_{6,i}); \ \mu_i = f_i \cdot \mathrm{sign}(z_{7,i}); \\ \rho_i &= f_i \cdot \mathrm{sign}(z_{8,i}); \ f_i = \mathrm{sign}(z_{1,i}); \end{aligned} \quad (8)$$

where $z_{j,i}$ denotes the jth component of a vector $\boldsymbol{Z}_i$ at the beginning of the (i+1)th iteration. For vectoring mode $\boldsymbol{Z}_i=\boldsymbol{Y}_i$ ($\boldsymbol{Y}_0=\boldsymbol{X}$), and for rotation mode, the control signs can be obtained from a preliminary vectoring operation for some vector $\mathbf{Z}$ on-the-fly. The result of (7) is

$$kY = \left(\prod_{i=0}^{n} R_{8,i}\right)X \quad (9)$$

The hardware implementation of the OCA taken from Doukhnitch [2011]is shown in Fig. 1.

The processor is composed of shifters, which performs the multiplication using t_i, 7-to-2 Carry Save Adder (CSA) arrays, and 3-input full adders. The $S_1, S_2, \ldots, S_8$ at the top of the figure are the components where the required control signs for the next iteration are prepared, the details of which are given in (8). Owing to their ineffectiveness in the sense of execution time with respect to our design and to sustain the traceability of the figure, these components are shown separately. The control signs $\alpha_i \ldots \rho_i$ are used subsequently to affect the signs of the elements of 8-D vector Y_i. The I/O registers are designed such that the data transfer into and out of the processor can take place simultaneously. In Fig. 1, components labelled as −1 are used to change a sign of operand.

Quaternion Jacobi Transformation

In this paper, as a particular case, we cosider only Hermitian matrixes because they have own important applications though the offered approach can be extended to a case of rectangular matrixes. The Jacobi procedure problem is as follows. A 2×2 block of a Hermitian quaternion $m\times m$matrix $\boldsymbol{A}$ is given:

$$\boldsymbol{A_{rc}} = \begin{bmatrix} a_{rr} & \boldsymbol{a_{rc}} \\ \boldsymbol{a_{cr}} & a_{cc} \end{bmatrix}, \quad (10)$$

where quaternion (bold face symbol) $\boldsymbol{a_{cr}} = \bar{\boldsymbol{a}}_{\boldsymbol{rc}}$ and diagonal elements are real. The over bar denotes (complex or quaternion) conjugation. In Zhang [1997], it was shown that a quaternion Jacobi rotation is analogous to a complex Jacobi rotation.

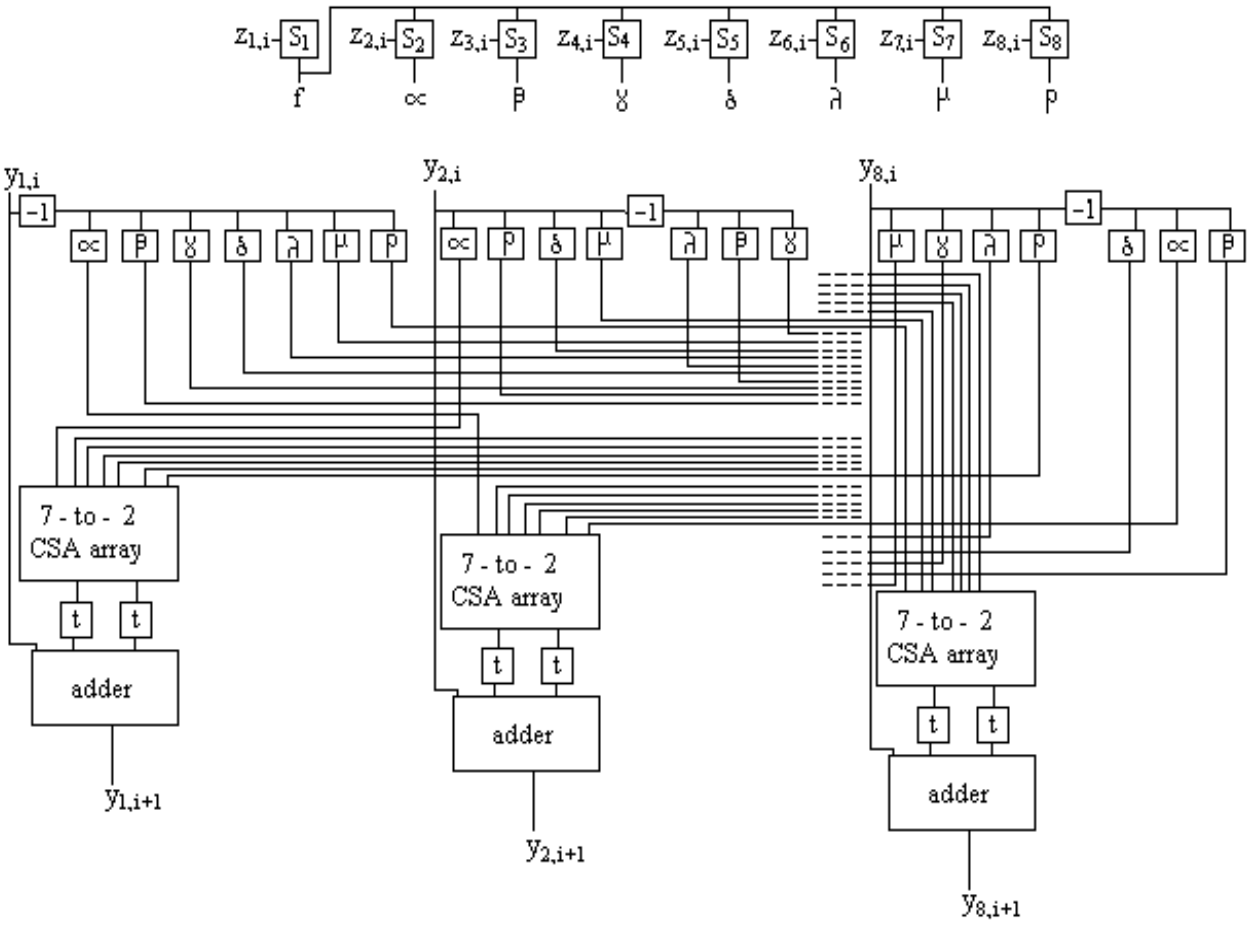

FIG. 1. OCTONION CORDIC-PROCESSOR ARCHITECTURE

Using a 2 × 2 quaternion unitarymatrix $\boldsymbol{P}$, we can perform a Jacobi rotation on $\boldsymbol{A}_{rc}$, i.e.

$$\boldsymbol{P}^{*}\boldsymbol{A}_{\boldsymbol{rc}}\boldsymbol{P} = \boldsymbol{R}, \tag{11}$$

where* stands for conjugation–transposition and $\boldsymbol{R}$ is a 2×2 quaternion diagonal matrix with real components only:

$$\boldsymbol{R} = \begin{bmatrix} r_{rr} & 0 \\ 0 & r_{cc} \end{bmatrix}$$

A general 2×2 unitary quaternion matrix $\boldsymbol{P}$ has elements which have moduli that are cosines and sines of some angle θ:

$$\boldsymbol{P} = \begin{bmatrix} c & \boldsymbol{s} \\ -\bar{\boldsymbol{s}} & \bar{c} \end{bmatrix} \tag{12}$$

with|c| = cos θ, |s| = sin θ and the constraint

$c\bar{c} + \boldsymbol{s}\bar{\boldsymbol{s}}$=1. The elements (real c and quaternion $\boldsymbol{s}$) can be calculated as following Bihan [2007]:

$$c = \cos\theta = \frac{1}{\sqrt{1+t^2}}; \boldsymbol{s} = \boldsymbol{a}_{rc}\frac{\sin\theta}{|\boldsymbol{a}_{rc}|} \tag{13}$$

where

$$t = \frac{sign\tau}{|\tau| + \sqrt{1+\tau^2}};$$
$$\tau = \cot 2\,\theta = \frac{a_{cc} - a_{rr}}{2|\boldsymbol{a}_{\boldsymbol{rc}}|};\ \sin\theta = tc$$

Two-sided rotation (11) can be executed with two 8-D rotations in sequence but the angle θ of rotation must be calculated before. This angle can be represented in no evident form (in an *implicit* fashion)as a sequence of sets of control signs $\{\alpha_i \ldots \rho_i\}_\theta$ ($i = \overline{0, \mathrm{n}}$)(see Delosme [1990]). In general, for a Hermitian matrix, to calculate an angle $\theta = \operatorname{atan} y/x$ we can apply algorithm (7) in vectoring mode with quaternion vector $\boldsymbol{X}$= [$\boldsymbol{x}$, $\boldsymbol{y}$]$^{\mathrm{T}}$ and the corresponding sequence will be generated. If we take $\boldsymbol{x}$=a_{jj}-a_{cc} and $\boldsymbol{y}$= $\boldsymbol{a}_{cr}$+$\overline{\boldsymbol{a}_{\boldsymbol{rc}}}$ we can calculate a sequence of sets $\{\alpha_i \ldots \rho_i\}_{2\theta}$ ($i = \overline{0, \mathrm{n}}$) corresponding to angle 2θ.

To determine the angle θ, this sequence can be transformed using a formula:

$$\tan\frac{\alpha}{2} = \frac{\sin\alpha}{1+\cos\alpha} \tag{14}$$

First, this sequence is applied to rotate an unit vector $\boldsymbol{X}$= [*1, 0, 0, 0, 0, 0, 0, 0*]$^{\mathrm{T}}$ to get two quaternion components of matrix (10) in the result $\boldsymbol{Y}$= [c, $\boldsymbol{s}$]$^{\mathrm{T}}$and then vectoring mode is utilized for modified vector

$$\boldsymbol{X}= [c+1, \boldsymbol{s}]^{\mathrm{T}} \tag{15}$$

As a result, we will get a sequence of sets $\{\alpha_i \ldots \rho_i\}_\theta$ ($i = \overline{0, \mathrm{n}}$) corresponding to angle θ. This sequence can be used for orthogonal transformation of matrix $\boldsymbol{A}$.

To speed up the considered calculations, we can use Two Plane Rotation (TPR) method to perform 2×2 SVD as two rotations in parallel suggested in Yang [1991]. This method represents 2×2 matrix (10) as a sum:

$$\boldsymbol{A}_{\boldsymbol{rc}} = \boldsymbol{A}_1 + \boldsymbol{A}_2 = \begin{bmatrix} p_1 & -\boldsymbol{q}_1 \\ \boldsymbol{q}_1 & p_1 \end{bmatrix} + \begin{bmatrix} -p_2 & \boldsymbol{q}_2 \\ \boldsymbol{q}_2 & p_2 \end{bmatrix}, \tag{16}$$

where p_1=(a_{rr}+a_{cc})/2, $\boldsymbol{q}_1$=($\boldsymbol{a}_{cr}$- $\overline{\boldsymbol{a}_{\boldsymbol{rc}}}$)/2, p_2=(a_{cc}-a_{rr})/2, $\boldsymbol{q}_2$=($\boldsymbol{a}_{cr}$+$\overline{\boldsymbol{a}_{\boldsymbol{rc}}}$)/2.

In general case, the transformation (11) can be executed as two rotations in vectoring mode for quaternion vectors $\boldsymbol{X}_1$=($\boldsymbol{p}_1$, $\boldsymbol{q}_1$)$^{\mathrm{T}}$ and $\boldsymbol{X}_2$=($\boldsymbol{p}_2$, $\boldsymbol{q}_2$)$^{\mathrm{T}}$. As a result, the matrix $\boldsymbol{R}$ is calculated as

$$\boldsymbol{R} = \begin{bmatrix} y_{11} - y_{21} & 0 \\ 0 & y_{11} + y_{21} \end{bmatrix},$$

where y_{11} and y_{21} are the first components of the resulting vectors $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ correspondently. In addition, two sequences of sets $\{\alpha_i \ldots \rho_i\}$ ($i = \overline{0, \mathrm{n}}$) corresponding to angles θ_+ and θ_- for TPR are produced. Then we can use two angles for two-sided rotation (11)$\theta_1 = (\theta_+ - \theta_-)/2$ and $\theta_2 = (\theta_+ + \theta_-)/2$ to modify off-diagonal elements of $\boldsymbol{A}$(two rows and two columns).

Actually, for Hermitian matrices, we have $\boldsymbol{q}_1$=0 and the first rotation is unnecessary (θ_-=0, y_{11}= p_1, and $\theta_1 = \theta_2 = \theta_+/2$). The elements p_1 and p_2 are real and $\boldsymbol{q}_2$ =$\overline{\boldsymbol{a}_{\boldsymbol{ij}}}$ is a quaternion. Therefore, we can use sets $\{\alpha_i \ldots \rho_i\}$ ($i = \overline{0, \mathrm{n}}$) corresponding to angle θ_+ to rotate an unit vector for generating vector (15). Then the sequence of sets $\{\alpha_i \ldots \rho_i\}_\theta$ ($i = \overline{0, \mathrm{n}}$) can be obtained corresponding to angle θ_1=θ_2 for rotation matrix $\boldsymbol{P}$ which is used for orthogonal transformation of m×m matrix $\boldsymbol{A}$.

SVD Processor Array

The Jacobi algorithm (Rutishauser[1971])consists of iteratively applying a basic 2×2 diagonalization formula (11) until the entire m× m matrix is diagonal. It works in several 'sweeps' until convergence is achieved. In each sweep, it rotates away all the non-zero off-diagonalelements. But every such rotation of course creates other non-zerooff-diagonal elements. It can be shown, however, that the sum of the absolute values of the off-diagonalelements is reduced in each sweep. More precisely, the Jacobi method has quadratic convergence. Convergence is achieved in *O(log m)*sweeps(Rutishauser [1971]).

The algorithm for hardware realization of Jacobi transformation can be described as follows:

Algorithm *JA*:

s=0; $\boldsymbol{A}^{(s)}$=$\boldsymbol{A}$;

$While \frac{SUM_1^{(s)}}{SUM_2^{(s)}} > \varepsilon$ {

For step =1 to m-1 {
Generation of m/2 pairs (r, c);
For each pair in parallel calculate rotation angles (sequence of sets{$\alpha_i \ldots \rho_i\}_\theta$);
rotate: $\boldsymbol{A}^{(s+1)} = \boldsymbol{P}^{(s)*}\boldsymbol{A}^{(s)}\boldsymbol{P}^{(s)}$ *-- usingTPR*
} –end for
s=s+1
} –end while

where SUM_1-sum of off-diagonal moduli of $\boldsymbol{A}^{(s)}$ in sweep *s*, SUM_2-sum of on-diagonal moduli of $\boldsymbol{A}^{(s)}$ in sweep *s*, matrix $\boldsymbol{P}^{(s)}$is a plan rotation in the (*r, c*) plane defined by the parameters(*c*,***s***,-***s***,*c*) in the (*rr*,*rc*,*cr*,*cc*) entries of an *m*×*m* identity matrix. For serial implementation, the simple strategy for generation of index pairs (*r, c*) is to chose the "cyclic-by-rows" ordering

(1,2), (1,3), ...,(1, *m*), (2,3),...,(*m-1, m*).

For parallel implementation, we can use "Brent-Luk" ordering (see Brent [1983], [1985]). A sweep for *(m-1)* steps has been executed with *m*/2 parallel annihilations per each.

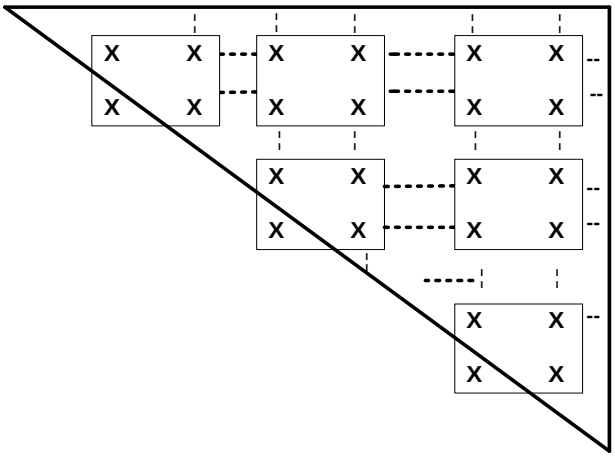

FIG. 2. TRIANGLE PROCESSOR ARRAY FOR JACOBI SVD

Parallel implementation of *m*×*m* matrix singular value decomposition is based on square BLV array with *m*/2 processor elements (PE) per side, where each processor contains one 2×2 submatrix, e.g. see Brent [1985]. Taking into account the equality $a_{rc} = \bar{a}_{cr}$ for Hermitian matrix, a triangle processor array can be used with one diagonal PE and *(m/2-1)* off-diagonal PEs per horizontal side, and one diagonal PE and *(m/2-1)* off-diagonal PEs per vertical side (see Brent [1985]). The total number of PEs is *m*/2diagonal PEs and (*m/2-1)*×*m/4*off-diagonal PEs. The configuration of this array is shown in Fig. 2, where crosses are matrix quaternion elements. All the PEs have nearest- neibour connections for permutations (17), and details are given in Brent [1985]). A phase of internal rearrangement of the elements of $\boldsymbol{A}_{rc}$ must be made after each step,see also Ma [1999]. Each diagonal PE is connected with all off-diagonal PEs in the row and in the column to send them the sequence of sets $\{\alpha_i \ldots \rho_i\}_\theta$ for rotations.

All array PEs are built using module for OCA implementation shown in Fig. 1. Each PE performs a Jacobi transformation for 2×2 matrix with quaternion elements. In Fig. 3 and Fig. 4, Octonion CORDIC application and evaluation symbols are shown correspondently.

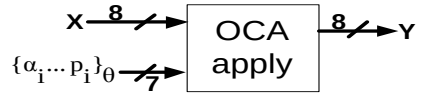

FIG. 3.OCTONION CORDIC APPLICATION

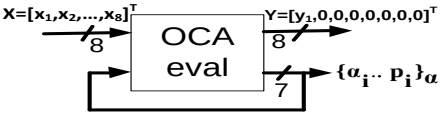

FIG. 4. OCTONION CORDIC EVALUATION

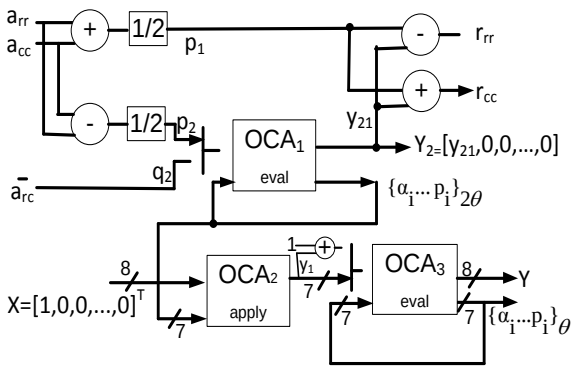

FIG. 5. DIAGONAL PE OF THE SVD ARRAY

Diagonal PE consists of 2 OCA modules described in Fig.1 (OCA_1 and OCA_3) to execute CORDIC evaluations (see Fig.5) and one module OCA_2 to rotate vector $\boldsymbol{X}$= [*1, 0, 0, 0, 0, 0, 0, 0*]^T to get two quaternion components of matrix (12) in the result $\boldsymbol{Y}$= [c, s]^T. This result is used to calculate vector (15) for module OCA_3 to generate evaluated control signs and to send them to the off-diagonal PEs in the same row and the same columnon-the-fly. As soon as the control signs are available, the off-diagonal processors can perform the CORDIC applications for 2×1 quaternion vector as shown in Fig.3, where

$$\boldsymbol{Y}=\boldsymbol{PX}, \text{for} \boldsymbol{X}= [\boldsymbol{a}_{rl}, \boldsymbol{a}_{cl}]^T,\ l=c+1,c+2,\ldots,m,$$
$$\text{and for } \boldsymbol{X}= [\boldsymbol{a}_{er}, \boldsymbol{a}_{ec}]^T,\ e=1,2,\ldots,r-1 \qquad (17)$$

However, to organize a parallel processing for all PEs we should use the TPR for off-diagonal PEs as well.

Each off-diagonal PE represents a submatrix

$$\boldsymbol{A_{2x2}} = \begin{bmatrix} \boldsymbol{a_{11}} & \boldsymbol{a_{12}} \\ \boldsymbol{a_{21}} & \boldsymbol{a_{22}} \end{bmatrix} \quad (18)$$

as a sum (16) and then executes TPR:

$$[\boldsymbol{r}_1, \boldsymbol{t}_1]^T = \mathrm{P}(\Theta_-)[\boldsymbol{p}_1, \boldsymbol{q}_1]^T,$$
$$[\boldsymbol{r}_1, \boldsymbol{t}_1]^T = \mathrm{P}(\Theta_+)[\boldsymbol{p}_2, \boldsymbol{q}_2]^T, \quad (19)$$

where $p_1=(a_{11}+a_{22})/2$, $q_1=(a_{21}-\overline{a_{12}})/2$, $p_2=(a_{22}-a_{11})/2$, $q_2=(a_{21}+\overline{a_{12}})/2$, $\Theta_+=\Theta_1+\Theta_2$, $\Theta_-=\Theta_2-\Theta_1$. Angle Θ_2 comes from column diagonal PE and the angle Θ_1 is comes from row diagonal PE. These angles should be represented as two parallel sequences of sets of control signs $\{\alpha_i \ldots \rho_i\}_\theta$ ($i=\overline{0,\mathrm{n}}$), and in each iteration I the OCA module should execute one rotation with set $\{\alpha_i \ldots \rho_i\}$ of Θ_2 and one rotation with set $\{\alpha_i \ldots \rho_i\}$ of Θ_1 changing or not all signs to opposite values for addition or for subtraction:

$$\{\alpha_i \ldots \rho_i\}\theta_{+/-} =$$

$$\{\alpha_{i2}, (\pm\alpha_{i1}) \ldots \rho_{i2}, (\pm\rho_{i1})\} \quad (i=\overline{0,\mathrm{n}}) \quad (20)$$

Therefore, all off-diagonal PEs spend twice time for rotation with scaling factor k^2.

Then the transformed matrix $\boldsymbol{A'_{2x2}}$ is obtained as:

$$\boldsymbol{a'_{11}} = \boldsymbol{r_1} - \boldsymbol{r_2}, \boldsymbol{a'_{12}} = -\boldsymbol{t_1} + \boldsymbol{t_2},$$
$$\boldsymbol{a'_{21}} = \boldsymbol{t_1} + \boldsymbol{t_2}, \boldsymbol{a'_{22}} = \boldsymbol{r_1} + \boldsymbol{r_2}.$$

The hardware implementation of off-diagonal PE is shown in Fig.6.

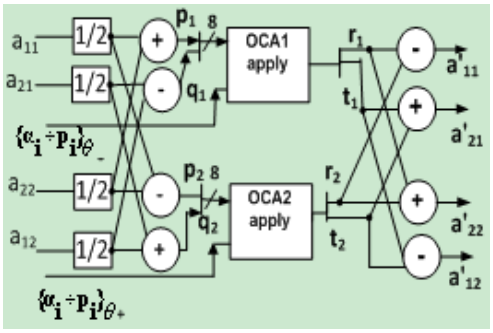

FIG. 6.OFF-DIAGONAL PE OF THE SVD ARRAY

Thus, the proposed algorithm provides the following levelsof processing:

- transformation ofall elements of the given matrix;
- overlapping in time of the rotation angle calculation androtation implementation;
- replacement of sequential two-sided rotations with two parallel one-sided rotations;
- simultaneous transformation of elements of 8-D vectors.

It is important to stress that the diagonal of resulting quaternion matrix has real components only.

The singular values and eigenvalues are the same for Hermitian matrices. So, the eigenvectors for the matrix $\boldsymbol{A}$, which will be identified by $\boldsymbol{B}$, can be obtained following the iterative process also proposed by Jacobi. Obtaining the matrix $\boldsymbol{B}$ is executed simultaneously with calculating the singular values (see Bravo [2008]). The matrix of eigenvectors (columns of $\boldsymbol{B}$) associated to the eigenvalues of $\boldsymbol{A}$, is also based on the decomposition of the matrix in submatrices of 2×2 elements. In this case, the input matrix to be subdivided is the identity matrix. Successive iterations will be applied, until the eigenvectors associated are found. To implement them, another (the second) processor array of size $m/2\times m/2$ can be used with no difference between diagonal PEs and off-diagonal PEs. The equation to compute for each processor is the following:

$$\boldsymbol{B}_{rc}^{(s+1)} = \boldsymbol{B}_{rc}^{(s)} \boldsymbol{P}_{rc}^{(s)}, \quad (21)$$

where $\boldsymbol{P}_{rc}^{(s)}$ is a matrix (12) and $\boldsymbol{B}_{rc}^{(s)}$ corresponds to systolic eigenvector processor which performs the CORDIC applications for two quaternion vectors with two slight modified OCA modules as in Fig. 7.

The eigenvector PE executes the left-side transformation instead of (21) as follows:

$$\boldsymbol{C}_{rc}^{(s+1)} = \boldsymbol{P}_{rc}^{(s)} \boldsymbol{C}_{rc}^{(s)},$$

where $\boldsymbol{C}_{rc} = \begin{bmatrix} \boldsymbol{b}_{rr} & -\boldsymbol{b}_{rc} \\ -\boldsymbol{b}_{cr} & \boldsymbol{b}_{cc} \end{bmatrix}$. The result of (21) can be achieved with sign changing for elements $\boldsymbol{b}_{rc}$ and $\boldsymbol{b}_{cr}$.

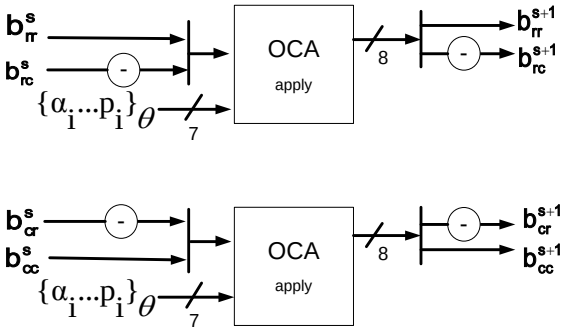

FIG. 7. EIGENVECTOR PE OF THE SVD ARRAY

The sequence of control signs for rotations is taken from corresponding DPEs of the first processor array in the same row on-the-fly. After each iteration, a process of rearrangement of the elements of each $\boldsymbol{B}_{rc}^{(s)}$ is made similar to the calculation of eigenvalues. After the same number of iterations as that in the eigenvalues calculation, the columns of resulting matrix $\boldsymbol{B}$ are the eigenvectors associated to the eigenvalues determined by the first array.

Error Analysis and Simulation Results

Following Hu [1992] and Paul [1995], we can analyse two types of quantization error: an approximation error due to the quantized representation of rotation angle and a rounding error due to the finite precision representation in fixed point arithmetic.

For fixed point arithmetic, all numbers have to be restricted to -1< x <1. Again, the number possesses h digits and machine accuracy is given by $\varepsilon_{Fi} = 2^{-h-1}$. If the result of the fixed point computation lies within the range, $z = Fi(x \pm y) = x \pm y$ holds (Paul [1995]). We use only operations $op \in \{+,-, shift\}$ calculated with a rounding error according to $z = Fi(x\ op\ y) = x\ op\ y + \varepsilon_{Fi}$. Therefore, if an error vector (see Hu [1992]) in each step due to rounding is

$$e(i) = [e_{1i}, e_{2i}, e_{3i}, e_{4i}, e_{5i}, e_{6i}, e_{7i}, e_{8i}]^T,$$

an upper bound for its absolute rounding error is:

$$|e(i)| \le \sqrt{8}\varepsilon_{Fi} = 2\sqrt{2}\varepsilon_{Fi} \quad (22)$$

Rotation angle θ is calculated as a sequence of sets $\{\alpha_i \ldots \rho_i\}_\theta$ ($i = \overline{0, n}$) with approximation error δ which is bounded by the smallest rotation angle σ. That is:

$$|\delta| \le \sigma = atan2^{-n} \quad (23)$$

The quantization error of an operation of vector rotation is governed by (seeHu [1992]):

$$\tilde{Y} = \left(\prod_{i=0}^{n-1} R_{8,i}\right) X + \sum_{i=0}^{n-1}\left(\prod_{j=i}^{n-1} R_{8,j}\right) e(i) + e(n) \quad (24)$$

Scaling correction introduces an additive error (Paul [1995]):

$$\hat{Y} = (1/k)\tilde{Y} + E\varepsilon_{Fi}, \quad (25)$$

where $E = [1, 1, 1, 1, 1, 1, 1, 1]^T$. Taking into account (22), the worst bound of (25) can be found (seeHu [1992]) with

$$\left\| k\hat{Y} - \left(\prod_{i=0}^{n-1} R_{8,i}\right) X \right\|_2 \le 2\sqrt{2}\varepsilon_{Fi} G(n), \quad (26)$$

where

$$G(n) = \left(1 + \sum_{i=0}^{n-1}\prod_{j=i}^{n-1}\|R_{8j}\|_2\right) = 1 + \sum_{i=0}^{n-1}\prod_{j=i}^{n-1}\sqrt{1 + 7 * 2^{-2j}}$$

For $m \times m$ matrices $\boldsymbol{A}$ and $\boldsymbol{P}$, the 2-norm of the absolute error of one sweep of SVD calculation is given (following Paul [1995]) by:

$$\|A^{(s)} - P^{(s)*} A^{(0)} P^{(s)}\|_2 \le \sqrt{2}\varepsilon_{Fi} G(n)\sqrt{(2m-4)}m(m-1) \quad (27)$$

and an overall error for p sweeps holds:

$$\left\| \Lambda - P^{*} A^{(0)} P \right\|_2 \le p\varepsilon_{Fi} G(n)\sqrt{(2m-4)}m(m-1)/2, \quad (28)$$

where $\boldsymbol{\Lambda}$–resulting diagonal matrix containing eigenvalues of $\boldsymbol{A}$. It is necessary to note that formulas (27) and (28) can be used with assumption that accuracy impacts of "cyclic-by-rows" ordering and "Brent-Luk" ordering are the same.

Simulation was executed using C# in Visual Studio 2010,. NET Framework 4.0 environment. Fig. 8 illustrates the evaluation mode of OCA (Fig. 4) as an average result of $E_e = \log_{10}(\sum_{j=2}^{8} |y_{ji}|/|y_{1i}|)$ for 100 pairs of quaternions with random elements in range ±1 where $y_{j,i}$ denotes the jth component of a vector $\boldsymbol{Y}_i$ at the beginning of the (i+1)th iteration.

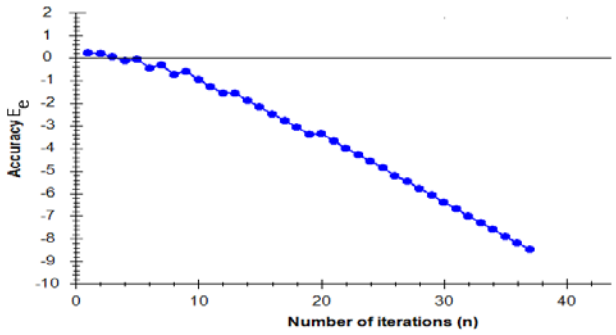

FIG. 8.ACCURACY OF OCA EVALUATION VS. NUMBER OF ITERATIONS

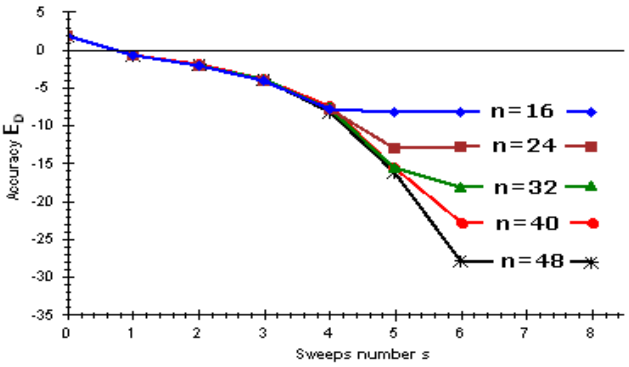

FIG. 9.ACCURACY OF DIAGONALIZATION VS. NUMBER OF SWEEPS

Following Bihan [2007], 1000 random Hermitian 15×15 quaternion matrices are taken for decomposition. In Fig.9, the diagonalization results are shown as an average value of $E_D = \log_{10}(S_1 / S_2)$ for n=16, 24, 32, 40,48.

Table 1 displays the simulation results in contrast to software implementation results of Jacobi SVD of arbitrary 15×15 quaternion matrices in Table 2 from

Bihan [2007], where RRE is the relative reconstruction error. Following Bihan [2007], in order to check the accuracy of the decomposition, we reconstruct the original matrix from the product (11), and calculate the mean relative modulus error between the elements of the original matrix and the reconstructed matrix. This is calculated as the mean over all matrix elements of $|q - q'| / 0.5|q + q'|$, where q is a quaternion element of the original matrix and q' is the corresponding element of the reconstructedmatrix. We also note the largest relative error across all elements of the matrix.

The comparison demonstrates that the Jacobi OCA implementation with the proposed processor array architecture needs almost the same number of sweeps as that for software implementation.

TABLE 1. SIMULATION RESULTS.

Parameter	Jacobi OCA	Quaternion Jacobi (Bihan [2007])	Complex Jacobi (Bihan [2007])
Sweeps	6.1	6.4	6.6
Rotations	658	672	2869
RRE	1e-16	1.88e-15	2.37e-11

Conclusions

In this study, a novel algorithm for VLSI implementation of Jacobi SVD for quaternion valued matrices was presented. It has a potential of speeding up important matrix algorithms such as calculation of eigenvalues and eigenvectors for Hermitian matrices. This algorithm can be used for triangularization of non-Hermitian matrices before SVD calculation as well. A possibility to speed up the process using TPR method for quaternion matrices has been indicated. The simple way to calculate a half angle in no evident form (in an implicit fashion) has been developed. A new organization of Jacobi SVD processor array to implement directly a quaternion matrix diagonalization without its decomposition into equivalent complex matrix has been described. In addition to diagonalization, the calculation of eigenvectors with the second processor array in parallel has been shown. Error analysis of the suggested algorithm is also given. The simulation results have confirmed the possibility to use OCA with suggested modifications for Jacobi SVD.

It's crucial to mention that all improvements over the basic CORDIC processor (scaling iterations, redundant arithmetic, high-radix arithmetic, approximate CORDIC-based Jacobi algorithm (e.g. Gotze [1993], etc.) may be incorporated into the offered processor array as well. Processor arrays with hardware implementation of Jacobi OCA can be fruitful in many signal-processing problems requiring lots of parallel computations.

REFERENCES

Bihan, N. L., and S. J. Sangwine. "Jacobi Method for Quaternion Matrix Singular Value Decomposition." *Applied Mathematics and Computation* 187(2007): 1265-71.

Bravo, I. et al. "Novel HW Architecture Based on FPGAs oriented to Solve the Eigen Problem." *IEEE Trans. on VLSI Systems*12(2008): 1722-25.

Brent, R. P., F.T. Luk, and C. Van Loan."Computation of the Singular Value Decomposition Using Mesh-connected Processors", *Technical report, Cornell University, Ithaca, NY, USA (*1983).

Brent, R. P., F.T. Luk. "The Solution of Singular Value and Symmetric Eigenvalue Problems on Multiprocessor Arrays", *SIAM J. Sci. Statist. Comput., 6 (*1985): pp.69-84.

Cavallaro, J. R., and F.T. Luk."CORDIC Arithmetic for an SVD Processor." *Journal of Parallel and Distributed Computing*3(1988): 271 – 90.

Delosme, J.-M. "Bit-level Systolic Algorithm for the Symmetric Eigenvalue Problem." *Proc. Int. Conf. on Application Specific Array Processors, Princeton, N.J.*(1990): 770-81.

Doukhnitch, Evgueni, "Octonion CORDIC Algorithms for DSP." *Proc. 6th Symp.on DSP for Communication Systems, DSPCS'2002, Sydney, Australia*(2002): 159-63.

EmreOzen. "Hardware-oriented Algorithm for Quaternion Valued Matrix Decomposition." IEEE Transactions on Circuits and Systems--II: Express Briefs4(2011): 225-9.

Gotze, S. J., and P. M. Sauer. "An Efficient Jacobi-like Algorithm for Parallel Eigenvalue Computation." *IEEE Trans. on Computers* 9 (1993): 1058-63.

Hsiao, S. F., and Delosme, J. M. "Parallel Singular Value Decomposition of Complex Matrices Using Multidimensional CORDIC Algorithms." *IEEE Trans. on Signal Processing* 3 (1996): 685-97.

Hu, Y. H. "The Quantization Effects of the CORDIC Algorithm." *IEEE Transactions on Signal Processing* 4(1992): 834-44.

Janovska, D., and G. Opfer. "Givens Transformation Applied to Quaternion Valued Vectors." *BIT Numerical Mathematics,*

43(2003): 991–1002.

J.I. Mars. "Singular Value Decomposition of Quaternion Matrices: a New Tool for Vector-sensor Signal Processing." *Signal Processing* 7(2004): 1177–99.

Leo, S. D., G. Scolaric, and L. Solombrino. "Quaternionic Eigenvalue Problem." *Journal of Mathematical Physics* 43(2002): 5815–29.

Ma J., Parhi K. K., Deprettere E. F. "An Algorithm Transformation Approach to CORDIC Based Parallel Singular Value Decomposition Architectures." *Proc. Asilomar Conf. on Signals, Systems and Computers, Pacific Grove, CA*(1999): 1401-5.

Meher, P. et al. "50 Years of CORDIC: Algorithms, Architectures and Applications." *IEEE Trans. on Circuits and Systems I: Regular Papers* 9(2009): 1893 – 907.

Miron, S., N. L. Bihan, and J. I. Mars. "High Resolution Vector-sensor Array Processing Using Quaternions." *Proc. IEEE/SP 13th Workshop on Statistical Signal Processing* (2005): 918 – 23.

N.L. Bihan, and J.I. Mars. "Quaternion-MUSIC for Vector-sensor Array Processing." *IEEE Transactions on Signal Processing* 4(2006): 1218–29. et al. "Multidimensional Signal Processing Using Quaternions." *Proc. 3rd Workshop on Physics Signal Image Processing, Grenoble, France* (2003): 57–60.

Paul, S., J. Gotze, and M. Sauer. "Error Analysis of CORDIC-Based Jacobi Algorithms", *IEEE Trans. on Computers* 7 (1995): 947-51.

Pei, S. C., J. H. Chang, J., and J. J. Ding. "Quaternion Matrix Singular Value Decomposition and its Applications for Color Image Processing." *Proc. Int. Conf. Image Process.*1(2003): 805–808.

Rader, C. M. "VLSI systolic arrays for adaptive nulling." *IEEE Signal Processing Mag*. 4 (1996): 29-49.

Rutishauser H., Contribution II/1, in: *Handbook for Automatic Computation*, ed. J. H. Wilkinson, C. Reinsch, Springer, 1971.

Sangwine, S. J. "Fourier Transforms of Color Images Using Quaternions, or Hypercomplex Numbers." *Electronics Letters* 21(1996): 1979-80.

S.-F., J.-M. and Delosme. "Householder CORDIC Algorithms."*IEEE Trans. on Computers* 8(1995): 900-1002.

S.J. Sangwine. "Quaternion Principal Component Analysis of Color Images." *Proc. IEEE Int. Conf. Image Process. (ICIP)* 1(2003): 809–12.

Snopce, H., and I. Spahiu. "Parallelization of SVD of a Matrix-systolic Approach." *Proc. of the 2010 International Multiconference on Computer Science and Information Technology (IMCSIT*)(2010): 343 –48.

Strumpen, V., H. Hoffmann, and A. Agarwal. "A Stream Algorithm for the SVD." *Technical report, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology* (2003).

Ward, J. P. *Quaternions and Cayley Numbers*, Kluwer Academic Publishers, 1997.

Yang, B., and J. F. Böhme. "Reducing the Computations of the Singular Value Decomposition Array Given by Brent and Luk." *SIAM J. Matrix Anal. Appl.*4(1991): 713–25.

Zhang, F. Z. "Quaternions and Matrices of Quaternions." *Linear Algebra Appl.* 251(1997): 21–57.

Evgueni Doukhnitch received PhD and DSc from Taganrog State Radio-Technical University, Russia. In 1999-2010, he was with the Department of Computer Engineering, Eastern Mediterranean University, Northern Cyprus, as a Professor. Now he is a Professor at Computer Engineering Department, Istanbul Aydin University, Turkey. His research interests are in the areas of hardware-oriented algorithms, hardware realization of linear algebra problems, and special-purpose processors.

Vadim Podbelskiy received PhD and DSc from Moscow Engineering Physic University, Russia. Now he is a Professor at Software Engineering Department of the Moscow National Research University-Higher School of Economics, Russia. His research interests are in the areas of applied mathematics, and simulation systems.